ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА

по дисциплине «Технологии разработки программного обеспечения»
на тему "QuizRunner. Система проведения тестирования."

Выполнил:
ст. гр. ИП-314
Бокий В.О

Проверил:
ст. преподаватель Токмашева Е. И.

Новосибирск, 2024

Оглавление

Введение и постановка задачи

Создать программу, в которой пользователь проходит тестирование и его уровень знаний оценивается программой.

Задачи:

1. Разработать main в котором вызываются функции всех алгоритмов программы.
2. Выбрать темы тестирования и подобрать тематические вопросы.
3. Разработать для каждого вопроса отдельную функцию.
4. Разработать общую логическую функцию для всех вопросов по определенной теме.

Техническое задание

QuizRunner. Система проведения тестирования.

О проекте

QuizRunner - это программа в виде теста, которая оценит ваши знания в той или иной области. Язык программирования - C++.

Принцип работы

Пользователь должен ответить на вопросы, которые выводит на экран программа. Перед началом, пользователь должен выбрать тему вопросов. После завершения теста, программа выводит количество правильных и неправильных ответов и уровень знаний в выбранной им темой. Также задается количество вопросов по данной теме. Для каждой темы эти значения отличаются.

Функциональность проекта

- На главном экране находится меню, в котором пользователь может выбрать **ИГРАТЬ** либо **ВЫХОД**. После нажатия кнопки **ИГРАТЬ**, будет меню с выбором тем опроса.

- Количество тем: 3.

- Первая тема вопросов: Кто этот футболист.
  Включает в себя 10 вопросов. Количество вариантов ответов 4. В конце опроса, показывается сколько правильных ответов было дано.

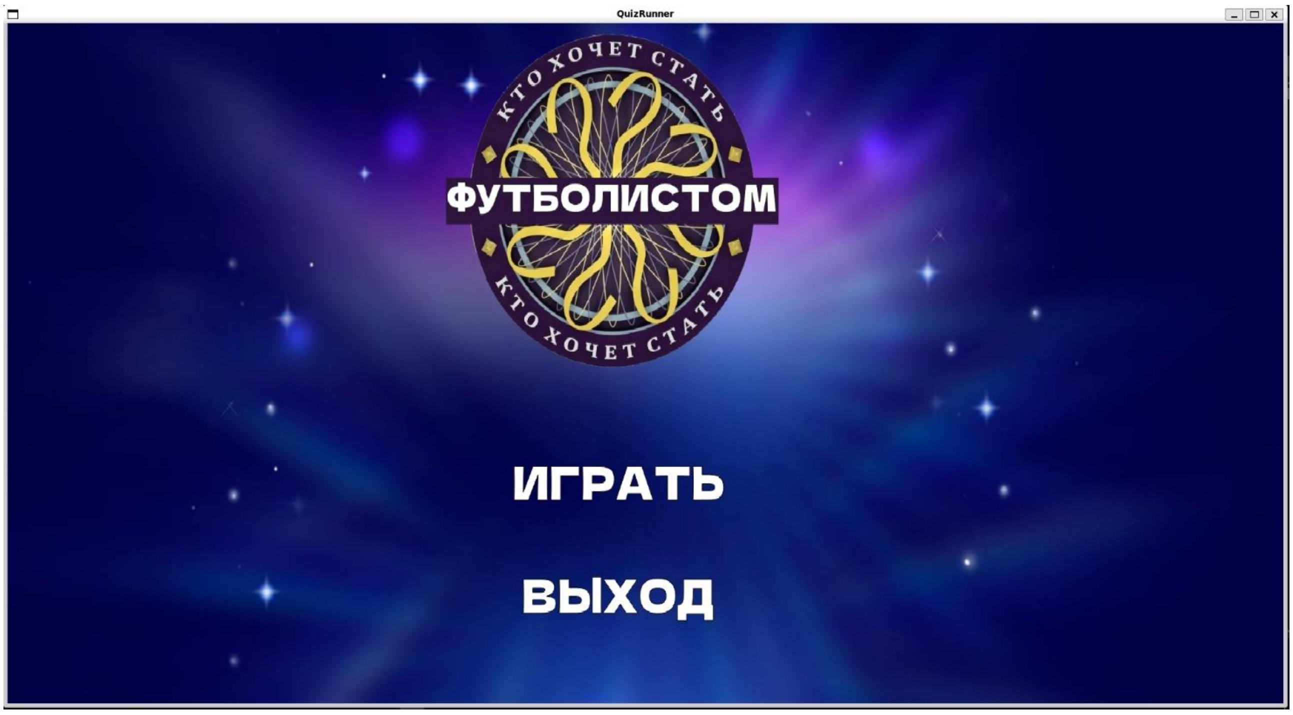
- Вторая тема вопросов: Какая это команда.
  Включает в себя также 10 вопросов. Количество вариантов ответов 4. В конце опроса, показывается сколько правильных ответов было дано.

- Третья тема вопросов: Откуда этот футболист.
  Включает в себя также 10 вопросов. Количество вариантов ответов 4. В конце опроса, показывается сколько правильных ответов было дано.

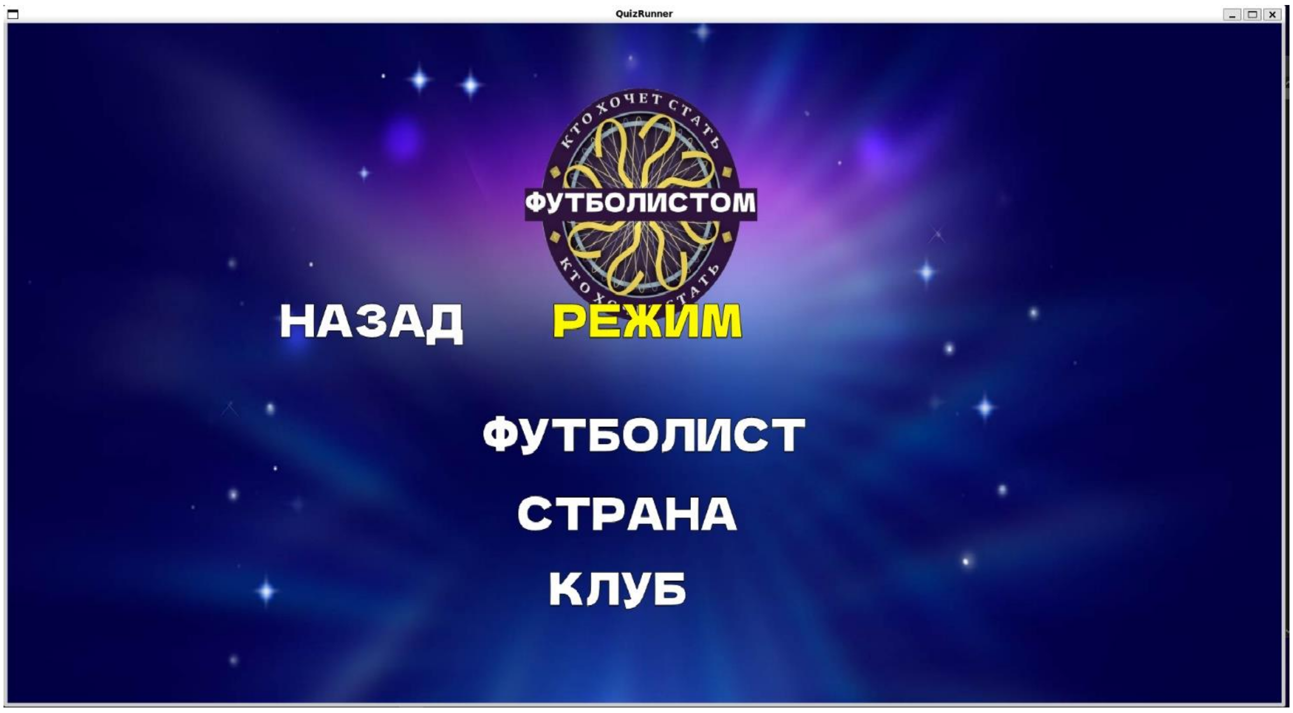
- Во время теста выводится спрайты и варианты ответов на этот вопрос, из предложенных вариантов необходимо выбрать один из вариантов при помощи курсора.

Описание выполненного проекта

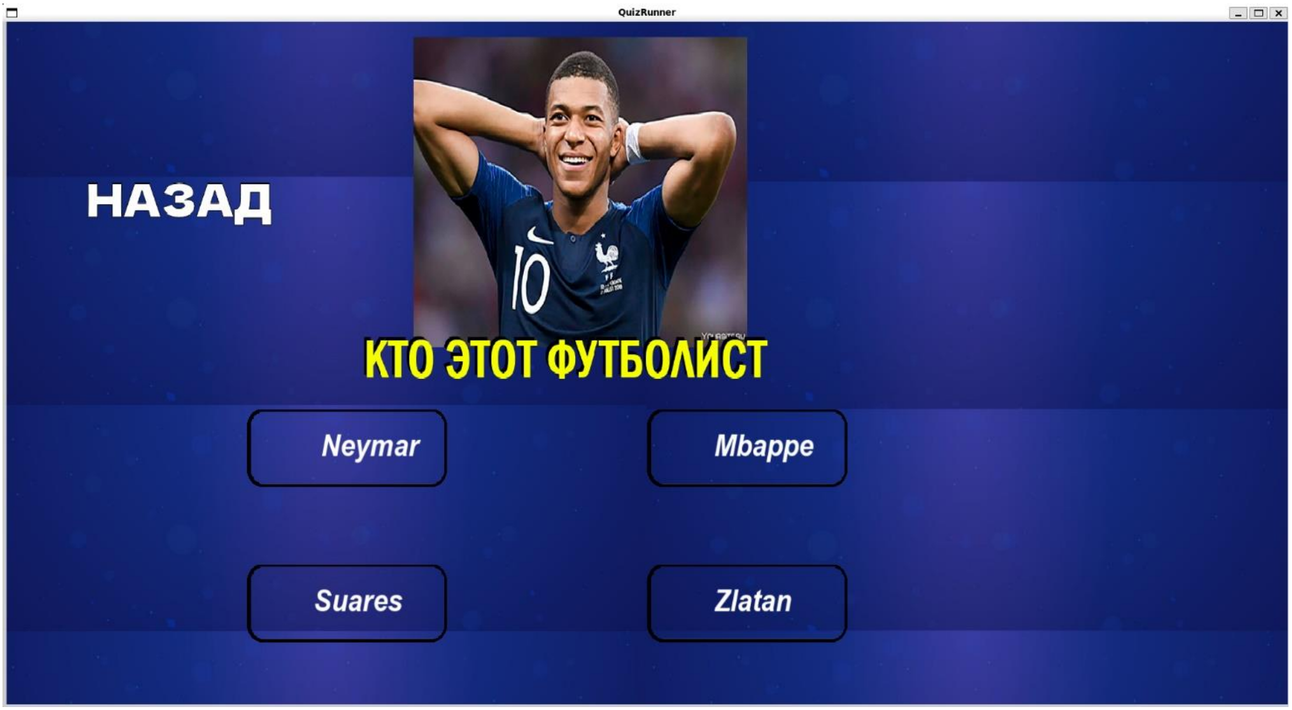
Главное меню программы:



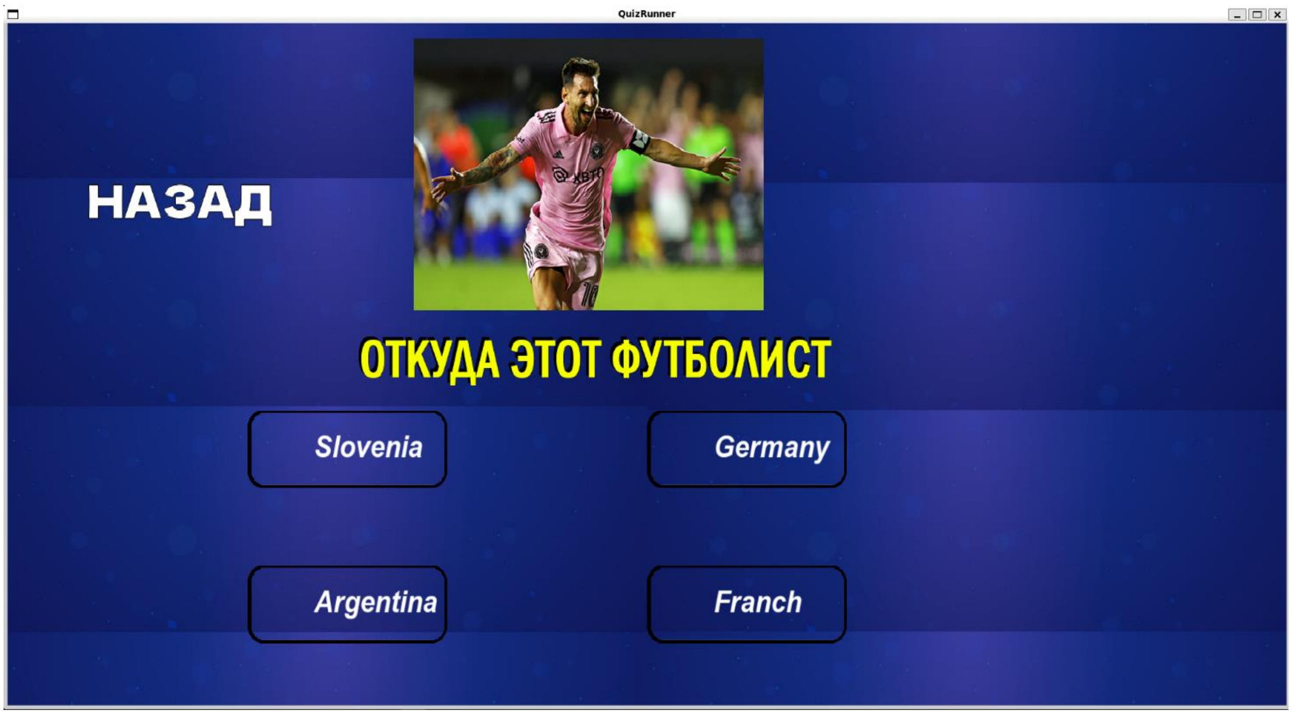
Выбор режимов:



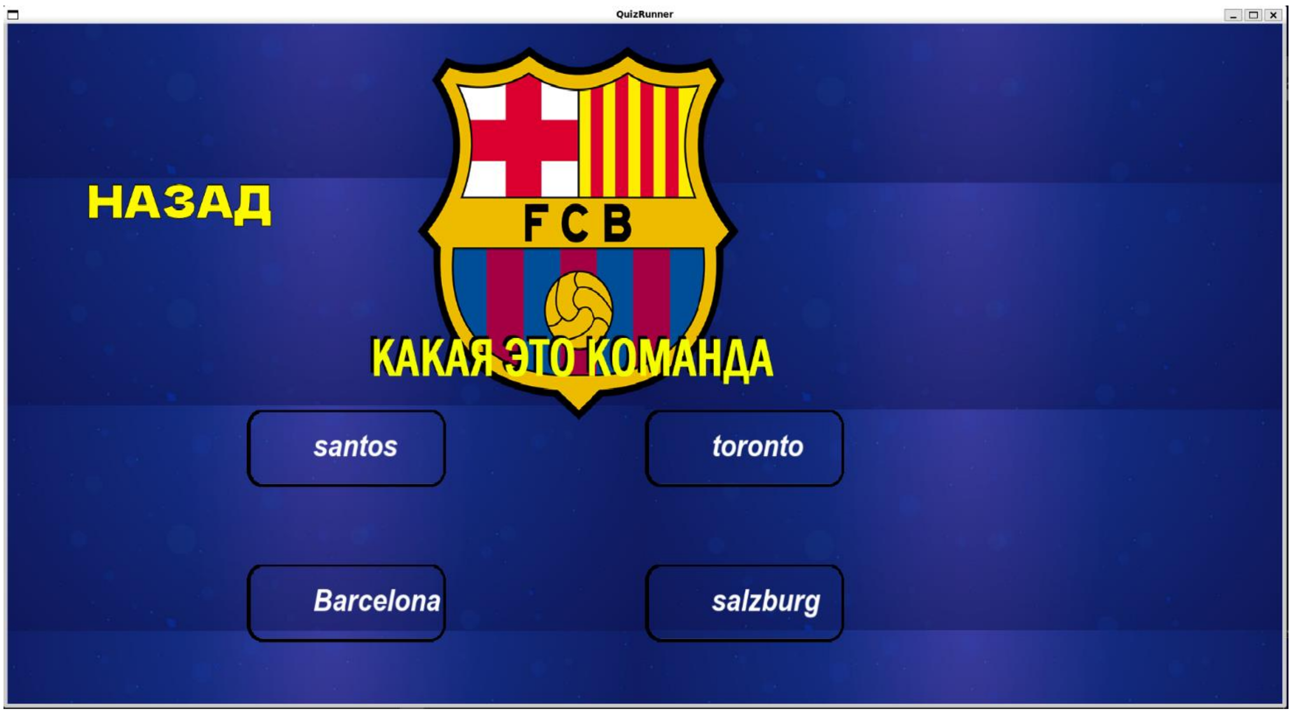
Режим “Футболист”:



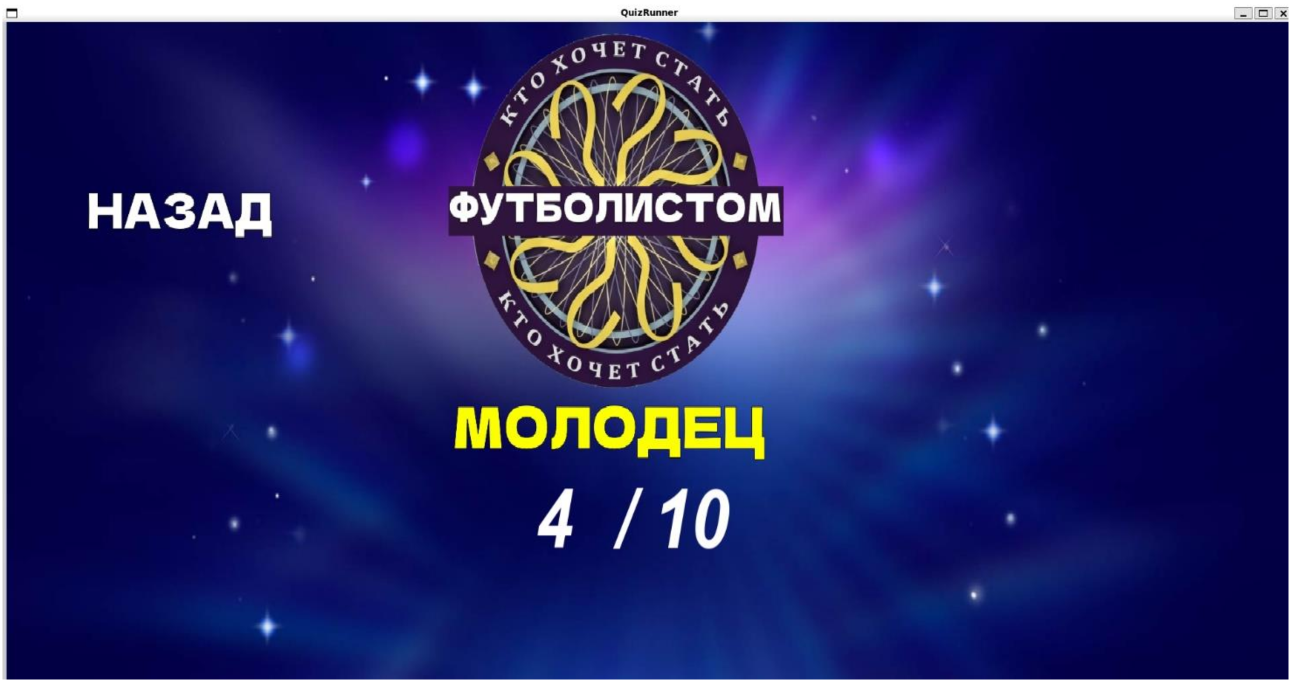
Режим “Страна”:



Режим “Клуб”

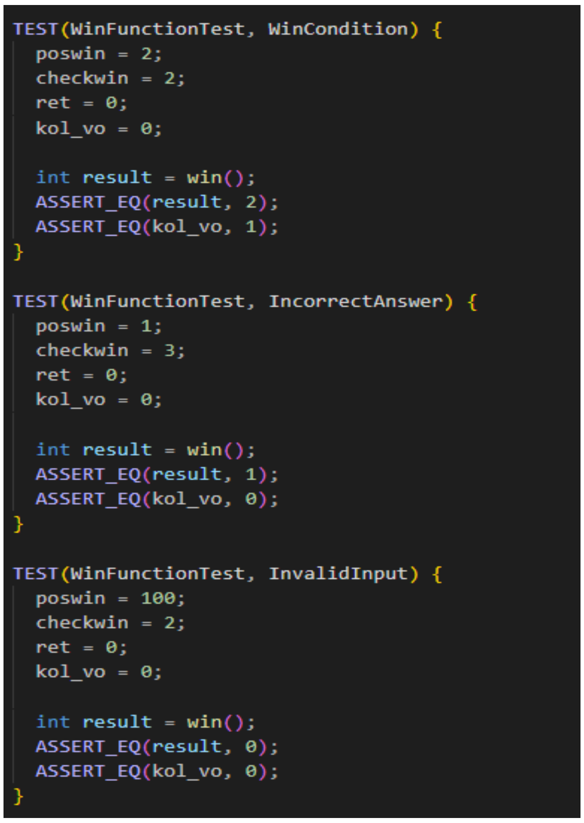


Результат в конце игры:



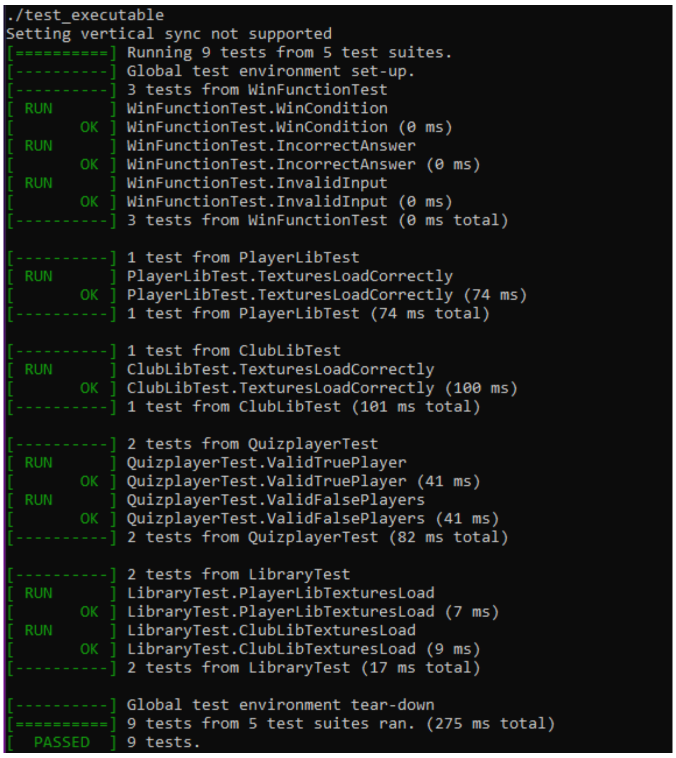
Тесты функции

Пример части кода тестов функции. В этом примере, тесты проверяют правильность работы функции win().



```cpp
TEST(WinFunctionTest, WinCondition) {
    poswin = 2;
    checkwin = 2;
    ret = 0;
    kol_vo = 0;

    int result = win();
    ASSERT_EQ(result, 2);
    ASSERT_EQ(kol_vo, 1);
}

TEST(WinFunctionTest, IncorrectAnswer) {
    poswin = 1;
    checkwin = 3;
    ret = 0;
    kol_vo = 0;

    int result = win();
    ASSERT_EQ(result, 1);
    ASSERT_EQ(kol_vo, 0);
}

TEST(WinFunctionTest, InvalidInput) {
    poswin = 100;
    checkwin = 2;
    ret = 0;
    kol_vo = 0;

    int result = win();
    ASSERT_EQ(result, 0);
    ASSERT_EQ(kol_vo, 0);
}
```

Выполнение тестов:



```
./test_executable
Setting vertical sync not supported
[==========] Running 9 tests from 5 test suites.
[----------] Global test environment set-up.
[----------] 3 tests from WinFunctionTest
[ RUN      ] WinFunctionTest.WinCondition
[       OK ] WinFunctionTest.WinCondition (0 ms)
[ RUN      ] WinFunctionTest.IncorrectAnswer
[       OK ] WinFunctionTest.IncorrectAnswer (0 ms)
[ RUN      ] WinFunctionTest.InvalidInput
[       OK ] WinFunctionTest.InvalidInput (0 ms)
[----------] 3 tests from WinFunctionTest (0 ms total)

[----------] 1 test from PlayerLibTest
[ RUN      ] PlayerLibTest.TexturesLoadCorrectly
[       OK ] PlayerLibTest.TexturesLoadCorrectly (74 ms)
[----------] 1 test from PlayerLibTest (74 ms total)

[----------] 1 test from ClubLibTest
[ RUN      ] ClubLibTest.TexturesLoadCorrectly
[       OK ] ClubLibTest.TexturesLoadCorrectly (100 ms)
[----------] 1 test from ClubLibTest (101 ms total)

[----------] 2 tests from QuizplayerTest
[ RUN      ] QuizplayerTest.ValidTruePlayer
[       OK ] QuizplayerTest.ValidTruePlayer (41 ms)
[ RUN      ] QuizplayerTest.ValidFalsePlayers
[       OK ] QuizplayerTest.ValidFalsePlayers (41 ms)
[----------] 2 tests from QuizplayerTest (82 ms total)

[----------] 2 tests from LibraryTest
[ RUN      ] LibraryTest.PlayerLibTexturesLoad
[       OK ] LibraryTest.PlayerLibTexturesLoad (7 ms)
[ RUN      ] LibraryTest.ClubLibTexturesLoad
[       OK ] LibraryTest.ClubLibTexturesLoad (9 ms)
[----------] 2 tests from LibraryTest (17 ms total)

[----------] Global test environment tear-down
[==========] 9 tests from 5 test suites ran. (275 ms total)
[  PASSED  ] 9 tests.
```

Личный вклад в проект

- Разработка Makefile
- Разработка тестов для приложения
- Разработка quiz.hpp и последующее подключение к другим частям кода
- Настройка CI

Приложение. Текст программы

```
//file quiz.cpp
#include <SFML/Graphics.hpp>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>

#include "quiz.hpp"

using namespace sf;
using namespace std;

RenderWindow window(VideoMode(w, h - 200), "QuizRunner");

int win()
{
    if ((poswin != 100) && (checkwin == poswin)) {  ret = 2; kol_vo++; }

    if ((poswin != 100) && (checkwin != poswin)) { ret =  1; }

    return ret;
}

void fonwin(Vector2f& pos)
{
    Font arial;
    arial.loadFromFile("app/Arial/arial_bolditalicmt.ttf");
    wintexture.loadFromFile("app/images/win.png");
    winsprite.setTexture(wintexture);
    winsprite.setPosition(650,450);
    Text p1(k[kol_vo] , arial, 100);
    Text p2("/ 10", arial, 100);
    p1.setPosition(w / 2 -170, h/2);
    p2.setPosition(w / 2 - 50, h / 2); // поменять место
    window.clear();

    if (backsprite.getGlobalBounds().contains(pos.x, pos.y)) { back-
sprite.setColor(Color(255, 250, 0)); }
    else { backsprite.setColor(Color::White); }

    whosprite.setPosition(w / 2 - 300, -20);

    window.draw(Fonsprite);
    window.draw(whosprite);
    window.draw(winsprite);
    window.draw(backsprite);
    window.draw(p1);
    window.draw(p2);
    window.display();
}
void hod(Vector2f& pos)
{
    if (p0S.getGlobalBounds().contains(pos.x, pos.y)) { poswin = 0;  }
    if (p1S.getGlobalBounds().contains(pos.x, pos.y)) { poswin = 1; }
    if (p2S.getGlobalBounds().contains(pos.x, pos.y)) { poswin = 2; }
    if (p3S.getGlobalBounds().contains(pos.x, pos.y)) { poswin = 3; }
}

//lib's
```

```
void playerlib()
{
    messitexture.loadFromFile("app/players/messi.jpg");
    mbappetexture.loadFromFile("app/players/mbappe.jpg");
    noyertexture.loadFromFile("app/players/noyer.jpg");
    neymartexture.loadFromFile("app/players/neymar.jpg");

    akinfeevtexture.loadFromFile("app/players/akinfeev.jpg");
    bufontexture.loadFromFile("app/players/bufon.jpg");
    modrictexture.loadFromFile("app/players/modric.jpg");

    oblaktexture.loadFromFile("app/players/yan_oblak.jpg");
    suarestexture.loadFromFile("app/players/suares.jpg");
    zlatantexture.loadFromFile("app/players/zlatan.jpg");

    oblaksprite.setTexture(oblaktexture);
    suaressprite.setTexture(suarestexture);
    zlatansprite.setTexture(zlatantexture);

    messiprite.setTexture(messitexture);
    mbappesprite.setTexture(mbappetexture);
    noyersprite.setTexture(noyertexture);
    neymarsprite.setTexture(neymartexture);

    akinfeevsprite.setTexture(akinfeevtexture);
    bufonsprite.setTexture(bufontexture);
    modricsprite.setTexture(modrictexture);

}
void clublib()
{
    BarcelonaT.loadFromFile("app/club/barcelona.png");
    BavariaT.loadFromFile("app/club/bayern-munich.png");
    BorissaT.loadFromFile("app/club/borussia.png");
    KobenhavnT.loadFromFile("app/club/kobenhavn.png");
    qingbaoT.loadFromFile("app/club/qingdao.png");
    RealT.loadFromFile("app/club/real-madrid.png");
    salzburgT.loadFromFile("app/club/salzburg.png");
    santosT.loadFromFile("app/club/santos.png");
    spartakT.loadFromFile("app/club/spartak.png");
    torontoT.loadFromFile("app/club/toronto.png");

    BarcelonaS.setTexture(BarcelonaT);
    BavariaS.setTexture(BavariaT);
    BorissaS.setTexture(BorissaT);
    KobenhavnS.setTexture(KobenhavnT);
    qingbaoS.setTexture(qingbaoT);
    RealS.setTexture(RealT);
    salzburgS.setTexture(salzburgT);
    santosS.setTexture(santosT);
    spartakS.setTexture(spartakT);
    torontoS.setTexture(torontoT);
}

//Quiz
void Quizplayer()
{
    playerlib();
    Sprite photos[Size] = { messiprite, mbappesprite, noyersprite, neymarsprite,
akinfeevsprite, bufonsprite, modricsprite, suaressprite,
zlatansprite,oblaksprite };

    int TrueP = rand() % 4;
    checkwin = TrueP;

    positions[TrueP] = 2;

    posnames[TrueP] = names[TruePlayer];
    p = photos[TruePlayer];

    int Falsep = rand() % 10;
    if (TrueP == 0)
    {
        posnames[1] = names[Falsep];
        posnames[2] = names[Falsep];
        posnames[3] = names[Falsep];
        while (posnames[0] == posnames[1]) { Falsep = rand() % 10; posnames[1] =
names[Falsep]; positions[1] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1])) {
Falsep = rand() % 10; posnames[2] = names[Falsep]; positions[2] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] =
names[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 1)
    {
        posnames[0] = names[Falsep];
        posnames[2] = names[Falsep];
        posnames[3] = names[Falsep];
        while (posnames[1] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
names[Falsep]; positions[0] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1])) {
Falsep = rand() % 10; posnames[2] = names[Falsep]; positions[2] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] =
names[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 2)
    {
        posnames[0] = names[Falsep];
        posnames[1] = names[Falsep];
        posnames[3] = names[Falsep];
        while (posnames[2] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
names[Falsep]; positions[0] = 1; }
        while ((posnames[1] == posnames[0]) || (posnames[1] == posnames[2])) {
Falsep = rand() % 10; posnames[1] = names[Falsep]; positions[1] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] =
names[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 3)
    {
        posnames[0] = names[Falsep];
        posnames[1] = names[Falsep];
        posnames[2] = names[Falsep];
        while (posnames[3] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
names[Falsep]; positions[0] = 1; }
        while ((posnames[1] == posnames[0]) || (posnames[1] == posnames[3])) {
Falsep = rand() % 10; posnames[1] = names[Falsep]; positions[1] = 1; }
```

```
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1]) ||
(posnames[2] == posnames[3])) { Falsep = rand() % 10; posnames[2] =
names[Falsep]; positions[2] = 1; }
    }

}
void Quizcountry()
{
    playerlib();

    Sprite photos[Size] = { messiprite, mbappesprite, noyersprite, neymarsprite,
akinfeevsprite, bufonsprite, modricsprite, suaressprite,
zlatansprite,oblaksprite };

    int TrueP = rand() % 4;
    checkwin = TrueP;

    positions[TrueP] = 2;

    posnames[TrueP] = country[TruePlayer];
    p = photos[TruePlayer];

    int Falsep = rand() % 10;
    if (TrueP == 0)
    {
        posnames[1] = country[Falsep];
        posnames[2] = country[Falsep];
        posnames[3] = country[Falsep];
        while (posnames[0] == posnames[1]) { Falsep = rand() % 10; posnames[1] =
country[Falsep]; positions[1] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1])) {
Falsep = rand() % 10; posnames[2] = country[Falsep]; positions[2] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] = coun-
try[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 1)
    {
        posnames[0] = country[Falsep];
        posnames[2] = country[Falsep];
        posnames[3] = country[Falsep];
        while (posnames[1] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
country[Falsep]; positions[0] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1])) {
Falsep = rand() % 10; posnames[2] = country[Falsep]; positions[2] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] = coun-
try[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 2)
    {
        posnames[0] = country[Falsep];
        posnames[1] = country[Falsep];
        posnames[3] = country[Falsep];
        while (posnames[2] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
country[Falsep]; positions[0] = 1; }
        while ((posnames[1] == posnames[0]) || (posnames[1] == posnames[2])) {
Falsep = rand() % 10; posnames[1] = country[Falsep]; positions[1] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] = coun-
try[Falsep]; positions[3] = 1; }
```

```
    }
    if (TrueP == 3)
    {
        posnames[0] = country[Falsep];
        posnames[1] = country[Falsep];
        posnames[2] = country[Falsep];
        while (posnames[3] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
country[Falsep]; positions[0] = 1; }
        while ((posnames[1] == posnames[0]) || (posnames[1] == posnames[3])) {
Falsep = rand() % 10; posnames[1] = country[Falsep]; positions[1] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1]) ||
(posnames[2] == posnames[3])) { Falsep = rand() % 10; posnames[2] = coun-
try[Falsep]; positions[2] = 1; }
    }

}
void Quizclub()
{
    //load
    clublib();
    //photos
    Sprite photos[Size] = { BarcelonaS, BavariaS, BorissaS, KobenhavnS, qing-
baoS, RealS, salzburgS, santosS, spartakS, torontoS };

    int TrueP = rand() % 4;
    checkwin = TrueP;

    positions[TrueP] = 2;

    posnames[TrueP] = club[TruePlayer];
    p = photos[TruePlayer];

    int Falsep = rand() % 10;
    if (TrueP == 0)
    {
        posnames[1] = club[Falsep];
        posnames[2] = club[Falsep];
        posnames[3] = club[Falsep];
        while (posnames[0] == posnames[1]) { Falsep = rand() % 10; posnames[1] =
club[Falsep]; positions[1] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1])) {
Falsep = rand() % 10; posnames[2] = club[Falsep]; positions[2] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] =
club[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 1)
    {
        posnames[0] = club[Falsep];
        posnames[2] = club[Falsep];
        posnames[3] = club[Falsep];
        while (posnames[1] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
club[Falsep]; positions[0] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1])) {
Falsep = rand() % 10; posnames[2] = club[Falsep]; positions[2] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] =
club[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 2)
    {
        posnames[0] = club[Falsep];
        posnames[1] = club[Falsep];
        posnames[3] = club[Falsep];
        while (posnames[2] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
club[Falsep]; positions[0] = 1; }
        while ((posnames[1] == posnames[0]) || (posnames[1] == posnames[2])) {
Falsep = rand() % 10; posnames[1] = club[Falsep]; positions[1] = 1; }
        while ((posnames[3] == posnames[0]) || (posnames[3] == posnames[1]) ||
(posnames[3] == posnames[2])) { Falsep = rand() % 10; posnames[3] =
club[Falsep]; positions[3] = 1; }
    }
    if (TrueP == 3)
    {
        posnames[0] = club[Falsep];
        posnames[1] = club[Falsep];
        posnames[2] = club[Falsep];
        while (posnames[3] == posnames[0]) { Falsep = rand() % 10; posnames[0] =
club[Falsep]; positions[0] = 1; }
        while ((posnames[1] == posnames[0]) || (posnames[1] == posnames[3])) {
Falsep = rand() % 10; posnames[1] = club[Falsep]; positions[1] = 1; }
        while ((posnames[2] == posnames[0]) || (posnames[2] == posnames[1]) ||
(posnames[2] == posnames[3])) { Falsep = rand() % 10; posnames[2] =
club[Falsep]; positions[2] = 1; }
    }

}

void mode (Vector2f& pos)
{
    mintexture.loadFromFile("app/images/whomin.png");
    minsprite.setTexture(mintexture);

    clubtexture.loadFromFile("app/images/club.png");
    clubsprite.setTexture(clubtexture);

    playertexture.loadFromFile("app/images/player.png");
    playersprite.setTexture(playertexture);

    countrytexture.loadFromFile("app/images/country.png");
    countrysprite.setTexture(countrytexture);

    backtexture.loadFromFile("app/images/back.png");
    backsprite.setTexture(backtexture);

    modetexture.loadFromFile("app/images/mode.png");
    modesprite.setTexture(modetexture);
    modesprite.setColor(Color(255, 250, 0));

    //position
    minsprite.setPosition(w / 2 - 180, 60);
    modesprite.setPosition(w / 2 - 170, 350);
    playersprite.setPosition(w/2 - 250, 500);
    countrysprite.setPosition(w / 2 - 200, 600);
    clubsprite.setPosition(w / 2 - 150, 700);
    backsprite.setPosition(w / 3 - 250, 355);


    if (playersprite.getGlobalBounds().contains(pos.x, pos.y)) { play-
ersprite.setColor(Color(255, 250, 0)); }
    else { playersprite.setColor(Color::White); }
    if (clubsprite.getGlobalBounds().contains(pos.x, pos.y)) { club-
sprite.setColor(Color(255, 250, 0)); }
    else { clubsprite.setColor(Color::White); }
    if (countrysprite.getGlobalBounds().contains(pos.x, pos.y)) { coun-
trysprite.setColor(Color(255, 250, 0)); }
    else { countrysprite.setColor(Color::White); }

    if (backsprite.getGlobalBounds().contains(pos.x, pos.y)) { back-
sprite.setColor(Color(255, 250, 0)); }
    else { backsprite.setColor(Color::White); }


    window.clear();

    window.draw(Fonsprite);
    window.draw(minsprite);
    window.draw(modesprite);
    window.draw(playersprite);
    window.draw(countrysprite);
    window.draw(clubsprite);
    window.draw(backsprite);

    window.display();
}
void menu(Vector2f& pos)
{

    Fontexture.loadFromFile("app/images/Fonmenu.jpg");
    Fonsprite.setTexture(Fontexture);
    whotexture.loadFromFile("app/images/who.png");
    whosprite.setTexture(whotexture);
    playtexture.loadFromFile("app/images/play.png");
    playsprite.setTexture(playtexture);
    exittexture.loadFromFile("app/images/exit.png");
    exitsprite.setTexture(exittexture);

    //position
    Fonsprite.setPosition(0, 0);
    whosprite.setPosition(w/2-300, -20);
    playsprite.setPosition(w/2 - 200, 560);
    exitsprite.setPosition(w/2 - 200, 710);

    if (playsprite.getGlobalBounds().contains(pos.x, pos.y)) { play-
sprite.setColor(Color(255,250,0)); }
    else { playsprite.setColor(Color::White); }
    if (exitsprite.getGlobalBounds().contains(pos.x, pos.y)) { ex-
itsprite.setColor(Color(255, 250, 0)); }
    else { exitsprite.setColor(Color::White); }

    window.clear();
    window.draw(Fonsprite);
    window.draw(whosprite);
    window.draw(playsprite);
    window.draw(exitsprite);
    window.display();
}

// Games
void player(Vector2f& pos)
{
    window.clear();
    window.draw(fullfonsprite);

    Font arial;
    arial.loadFromFile("app/Arial/arial_bolditalicmt.ttf");

    gamefontexture.loadFromFile("app/images/gamefon.png");
    gamefonsprite.setTexture(gamefontexture);
    fullfontexture.loadFromFile("app/images/fullfon.png");
    fullfonsprite.setTexture(fullfontexture);
    backtexture.loadFromFile("app/images/back.png");
    backsprite.setTexture(backtexture);
    gameT.loadFromFile("app/images/who is.png");

    pdT.loadFromFile("app/images/p2.png");

    p0T.loadFromFile("app/images/p.png");
    p0S.setTexture(p0T);
    p1S.setTexture(p0T);
    p2S.setTexture(p0T);
    p3S.setTexture(p0T);

    gameS.setTexture(gameT);

    gameS.setPosition(400,400);
    fullfonsprite.setPosition(0, 0);

    backsprite.setPosition(100, 200);
    if (backsprite.getGlobalBounds().contains(pos.x, pos.y)) { back-
sprite.setColor(Color(255, 250, 0)); }
    else { backsprite.setColor(Color::White); }

    Text p1(posnames[0], arial, 40);
    Text p2(posnames[1], arial, 40);
    Text p3(posnames[2], arial, 40);
    Text p4(posnames[3], arial, 40);

    p1.setPosition(w / 2 - 500, 520);
    p2.setPosition(w / 2 + 100, 520);
    p3.setPosition(w / 2 -500, 720);
    p4.setPosition(w / 2 + 100, 720);
    p.setPosition(w/2 - 350, 20);

    p0S.setPosition(w/2 - 600, 500);
    p1S.setPosition(w / 2 , 500);
    p2S.setPosition(w / 2 - 600, 700);
    p3S.setPosition(w / 2 , 700);

    if (p0S.getGlobalBounds().contains(pos.x, pos.y)) { p0S.setTexture(pdT); }
    else { p0S.setTexture(p0T); }
    if (p1S.getGlobalBounds().contains(pos.x, pos.y)) { p1S.setTexture(pdT); }
    else { p1S.setTexture(p0T); }
    if (p2S.getGlobalBounds().contains(pos.x, pos.y)) { p2S.setTexture(pdT); }
    else { p2S.setTexture(p0T); }
    if (p3S.getGlobalBounds().contains(pos.x, pos.y)) { p3S.setTexture(pdT); }
    else { p3S.setTexture(p0T); }

    if (names[TruePlayer] == "STOP") { m1 = 5;  TruePlayer = 0; }
    window.draw(p);
    if (Working == 1) { Quizplayer(); Working = 0; }
    if ((win() != 0))
    {
        window.draw(winsprite);
        poswin = 100;
        ret = 0;
        Working = 1;
        TruePlayer++;
    }

    window.draw(gameS);

    window.draw(p0S);
    window.draw(p1S);
    window.draw(p2S);
    window.draw(p3S);
    window.draw(p1);
    window.draw(p2);
    window.draw(p3);
    window.draw(p4);
    window.draw(backsprite);
    window.display();
}
void Club(Vector2f& pos)
{
    window.clear();
    window.draw(fullfonsprite);

    Font arial;
    arial.loadFromFile("app/Arial/arial_bolditalicmt.ttf");

    gamefontexture.loadFromFile("app/images/gamefon.png");
    gamefonsprite.setTexture(gamefontexture);
    fullfontexture.loadFromFile("app/images/fullfon.png");
    fullfonsprite.setTexture(fullfontexture);
    backtexture.loadFromFile("app/images/back.png");
    backsprite.setTexture(backtexture);

    gameT.loadFromFile("app/images/wich is.png");

    pdT.loadFromFile("app/images/p2.png");

    p0T.loadFromFile("app/images/p.png");
    p0S.setTexture(p0T);
    p1S.setTexture(p0T);
    p2S.setTexture(p0T);
    p3S.setTexture(p0T);

    fullfonsprite.setPosition(0, 0);

    gameS.setTexture(gameT);
    gameS.setPosition(400, 400);

    backsprite.setPosition(100, 200);
    if (backsprite.getGlobalBounds().contains(pos.x, pos.y)) { back-
sprite.setColor(Color(255, 250, 0)); }
    else { backsprite.setColor(Color::White); }

    Text p1(posnames[0], arial, 40);
    Text p2(posnames[1], arial, 40);
    Text p3(posnames[2], arial, 40);
    Text p4(posnames[3], arial, 40);
    p1.setPosition(w / 2 - 500, 520);
    p2.setPosition(w / 2 + 100, 520);
    p3.setPosition(w / 2 - 500, 720);
    p4.setPosition(w / 2 + 100, 720);
    p.setPosition(w/2 - 350, 20);

    p0S.setPosition(w / 2 - 600, 500);
    p1S.setPosition(w / 2, 500);
    p2S.setPosition(w / 2 - 600, 700);
    p3S.setPosition(w / 2, 700);

    if (club[TruePlayer] == "STOP") { m1 = 5;  TruePlayer = 0; }
    window.draw(p);
    if (Working == 1) { Quizclub(); Working = 0; }
    if ((win() != 0))
    {
        poswin = 100;
        ret = 0;
        Working = 1;
        TruePlayer++;
    }

    if (p0S.getGlobalBounds().contains(pos.x, pos.y)) { p0S.setTexture(pdT); }
    else { p0S.setTexture(p0T); }
    if (p1S.getGlobalBounds().contains(pos.x, pos.y)) { p1S.setTexture(pdT);}
    else { p1S.setTexture(p0T); }
    if (p2S.getGlobalBounds().contains(pos.x, pos.y)) { p2S.setTexture(pdT); }
    else { p2S.setTexture(p0T); }
    if (p3S.getGlobalBounds().contains(pos.x, pos.y)) { p3S.setTexture(pdT); }
    else { p3S.setTexture(p0T); }

    window.draw(gameS);

    window.draw(p0S);
    window.draw(p1S);
    window.draw(p2S);
    window.draw(p3S);
    window.draw(p1);
    window.draw(p2);
    window.draw(p3);
    window.draw(p4);
    window.draw(backsprite);
    window.display();
}

void Country(Vector2f& pos)
{
    window.clear();
    window.draw(fullfonsprite);

    Font arial;
    arial.loadFromFile("app/Arial/arial_bolditalicmt.ttf");

    gamefontexture.loadFromFile("app/images/gamefon.png");
    gamefonsprite.setTexture(gamefontexture);
    fullfontexture.loadFromFile("app/images/fullfon.png");
    fullfonsprite.setTexture(fullfontexture);
    backtexture.loadFromFile("app/images/back.png");
    backsprite.setTexture(backtexture);
    pdT.loadFromFile("app/images/p2.png");
    gameT.loadFromFile("app/images/where is.png");
    p0T.loadFromFile("app/images/p.png");
    p0S.setTexture(p0T);
    p1S.setTexture(p0T);
    p2S.setTexture(p0T);
    p3S.setTexture(p0T);

    fullfonsprite.setPosition(0, 0);

    backsprite.setPosition(100, 200);
    if (backsprite.getGlobalBounds().contains(pos.x, pos.y)) { back-
sprite.setColor(Color(255, 250, 0)); }
    else { backsprite.setColor(Color::White); }

    gameS.setTexture(gameT);
    gameS.setPosition(400, 400);


    Text p1(posnames[0], arial, 40);
    Text p2(posnames[1], arial, 40);
    Text p3(posnames[2], arial, 40);
    Text p4(posnames[3], arial, 40);

    p1.setPosition(w / 2 - 500, 520);
    p2.setPosition(w / 2 + 100, 520);
    p3.setPosition(w / 2 - 500, 720);
    p4.setPosition(w / 2 + 100, 720);
    p.setPosition(w/2 - 350, 20);

    p0S.setPosition(w / 2 - 600, 500);
    p1S.setPosition(w / 2, 500);
    p2S.setPosition(w / 2 - 600, 700);
    p3S.setPosition(w / 2, 700);

    if (p0S.getGlobalBounds().contains(pos.x, pos.y)) { p0S.setTexture(pdT); }
    else { p0S.setTexture(p0T); }
    if (p1S.getGlobalBounds().contains(pos.x, pos.y)) { p1S.setTexture(pdT); }
    else { p1S.setTexture(p0T); }
    if (p2S.getGlobalBounds().contains(pos.x, pos.y)) { p2S.setTexture(pdT); }
    else { p2S.setTexture(p0T); }
    if (p3S.getGlobalBounds().contains(pos.x, pos.y)) { p3S.setTexture(pdT); }
    else { p3S.setTexture(p0T); }

    window.draw(p);
    if (Working == 1) { Quizcountry(); Working = 0; }
    if ((win() != 0))
    {
        poswin = 100;
        ret = 0;
        Working = 1;
        TruePlayer++;


    }
    if (country[TruePlayer] == "STOP") { m1 = 5;  TruePlayer = 0; }

    window.draw(gameS);

    window.draw(p0S);
    window.draw(p1S);
    window.draw(p2S);
    window.draw(p3S);
    window.draw(p1);
    window.draw(p2);
    window.draw(p3);
    window.draw(p4);
    window.draw(backsprite);
    window.display();
}
int main()
{
    srand(time(NULL));
    window.setSize(Vector2u(w, h));

    Event event;
    while (window.isOpen())
    {

        Vector2i pixelPos = Mouse::getPosition(window);
        Vector2f pos = window.mapPixelToCoords(pixelPos);

        //vibor rezhima
        if (m1 == 0){ menu(pos); }
        if (m1 == 1) { mode(pos); }
        if (m1 == 2) { player(pos); }
        if (m1 == 3) { Country(pos); }
        if (m1 == 4) { Club(pos); }
        if (m1 == 5) { fonwin(pos); }


        while (window.pollEvent(event))
        {
            if (event.type == Event::Closed) { window.close(); }


            if (event.type == Event::MouseButtonPressed)
            {
                if (event.key.code == Mouse::Left)
                {

                    //menu
                    if (playsprite.getGlobalBounds().contains(pos.x, pos.y) &&
(m1 == 0)) { m1 = 1; }
                    if (backsprite.getGlobalBounds().contains(pos.x, pos.y) &&
(m1 == 1)) { m1 = 0; }
                    if (backsprite.getGlobalBounds().contains(pos.x, pos.y) &&
((m1 == 2) || (m1 == 3) || (m1 == 4) || (m1 == 5))) { m1 = 1; Working = 1; True-
Player = 0; kol_vo = 0; TruePlayer = 0; }
                    if (exitsprite.getGlobalBounds().contains(pos.x, pos.y) &&
(m1 == 0)) { window.close(); }
                    //mode
                    if (playersprite.getGlobalBounds().contains(pos.x, pos.y) &&
(m1 == 1)) { m1 = 2; }
                    if (countrysprite.getGlobalBounds().contains(pos.x, pos.y)
&& (m1 == 1)) { m1 = 3; }
                    if (clubsprite.getGlobalBounds().contains(pos.x, pos.y) &&
(m1 == 1)) { m1 = 4; }
                    //games
                    if ((m1 == 2) && (win() == 0)) { hod(pos); }
                    if ((m1 == 3) && (win() == 0)) { hod(pos); }
                    if ((m1 == 4) && (win() == 0)) { hod(pos); }
                }
            }
        }

    }
    return 0;
}
```

```
//file quiz.hpp

#ifndef QUIZ_LIB_HPP
#define QUIZ_LIB_HPP

#include <SFML/Graphics.hpp>
#include <iostream>

int poswin, checkwin, ret, positions[4];
int m1 = 0;
int TruePlayer = 0;
int Working = 1;
int kol_vo = 0;
const int Size = 11;
std::string k[11] = {"0","1", "2", "3", "4", "5", "6", "7", "8", "9", "10" };

float w = sf::VideoMode::getDesktopMode().width;
float h = sf::VideoMode::getDesktopMode().height;

float ttime;

sf::Texture wintexture;
sf::Sprite winsprite;

std::string names[Size] = {"Messi", "Mbappe", "Noyer", " Ney-
mar","Akinfeev","Bufon","Modric","Suares","Zlatan","Oblak","STOP"};
std::string country[Size] = { "Argentina", "Franch", "Germany", " Brazil","Rus-
sia","Italy","Croatia","Uruguay","Sweden","Slovenia","STOP"};
std::string club[Size] = { "Barcelona", "Bavaria", "Borissa", "Kobenhavn",
"qingbao", "Real Madrid", "salzburg", "santos", "spartak", "toronto","STOP" };
std::string posnames [4];

sf::Sprite photos[Size];
sf::Sprite p;

sf::Texture Fontexture, whotexture, playtexture, exittexture;
sf::Sprite Fonsprite, whosprite, playsprite, exitsprite;

sf::Texture mintexture, clubtexture, playertexture, countrytexture, modetexture,
backtexture;
sf::Sprite minsprite, clubsprite, playersprite, countrysprite, modesprite, back-
sprite;

sf::Texture gamefontexture, fullfontexture, p0T, p1T, p2T, p3T, pdT, gameT;
sf::Sprite gamefonsprite, fullfonsprite, p0S, p1S, p2S, p3S, gameS;

sf::Texture messitexture, mbappetexture, noyertexture, neymartexture, akinfeev-
texture, bufontexture, modrictexture, suarestexture, zlatantexture, oblaktex-
ture;
sf::Sprite messiprite, mbappesprite, noyersprite, neymarsprite, akinfeevsprite,
bufonsprite, modricsprite, suaressprite, zlatansprite, oblaksprite;

sf::Texture BarcelonaT, BavariaT, BorissaT, KobenhavnT, qingbaoT, RealT, salz-
burgT, santosT, spartakT, torontoT;
sf::Sprite BarcelonaS, BavariaS, BorissaS, KobenhavnS, qingbaoS, RealS, salz-
burgS, santosS, spartakS, torontoS;

int win();
void fonwin(sf::Vector2f& pos);
void hod(sf::Vector2f& pos);
void playerlib();
void clublib();
void Quizplayer();
void Quizcountry();
void Quizclub();
void mode (sf::Vector2f& pos);
void menu(sf::Vector2f& pos);
void player(sf::Vector2f& pos);
void Club(sf::Vector2f& pos);
void Country(sf::Vector2f& pos);

#endif
```

```
//makefile

CC = g++
CFLAGS = -Wall -Wextra -std=c++2a -pthread

SFML_DIR = external/SFML
SFML_INCLUDE = -I$(SFML_DIR)/include/SFML
SFML_LIBS = -lsfml-graphics -lsfml-window -lsfml-system

GTEST_DIR = external/googletest
GTEST_BUILD_DIR = $(GTEST_DIR)/build
GTEST_INCLUDE = -I$(GTEST_DIR)/googletest/include
GTEST_LIB = $(GTEST_BUILD_DIR)/lib/libgtest.a

SRC_DIR = app
TEST_DIR = app/test

SOURCES = $(wildcard $(SRC_DIR)/game/main.cpp)
OBJECTS = $(SOURCES:.cpp=.o)

TEST_SOURCES = $(wildcard $(TEST_DIR)/*.cpp)
TEST_OBJECTS = $(TEST_SOURCES:.cpp=.o)

EXECUTABLE = quiz
TEST_EXECUTABLE = test_executable

all: $(EXECUTABLE) $(TEST_EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	$(CC) $(CFLAGS) $(OBJECTS) -o $(EXECUTABLE) $(SFML_LIBS)

run: $(EXECUTABLE)
	./$(EXECUTABLE)

run_tests: $(TEST_EXECUTABLE)
	./$(TEST_EXECUTABLE)

$(TEST_EXECUTABLE): $(TEST_OBJECTS)
	$(CC) $(CFLAGS) $(TEST_OBJECTS) -o $(TEST_EXECUTABLE) $(GTEST_LIB)
$(SFML_LIBS)

%.o: %.cpp
	$(CC) $(CFLAGS) $(SFML_INCLUDE) $(GTEST_INCLUDE) -c $< -o $@

clean:
	rm -f $(EXECUTABLE) $(OBJECTS) $(TEST_EXECUTABLE) $(TEST_OBJECTS)

format:
	clang-format -i  $(SOURCES) $(TEST_SOURCES)
```

```
//test.cpp

#include "../game/quiz.hpp"
#include "../game/quiz.cpp"
#include <SFML/Graphics.hpp>
#include <cstdlib>
#include <ctime>
#include <gtest/gtest.h>

using namespace sf;

bool mockLoadFromFile(sf::Texture& texture, const std::string& filename)
{
    return texture.create(100, 100); // Создание фиктивной текстуры размером
100x100
}

// Переопределение функции загрузки текстуры для тестирования
void mockPlayerLib()
{
    mockLoadFromFile(messitexture, "app/players/messi.jpg");
    mockLoadFromFile(mbappetexture, "app/players/mbappe.jpg");
    mockLoadFromFile(noyertexture, "app/players/noyer.jpg");
    mockLoadFromFile(neymartexture, "app/players/neymar.jpg");
    mockLoadFromFile(akinfeevtexture, "app/players/akinfeev.jpg");
    mockLoadFromFile(bufontexture, "app/players/bufon.jpg");
    mockLoadFromFile(modrictexture, "app/players/modric.jpg");
    mockLoadFromFile(oblaktexture, "app/players/yan_oblak.jpg");
    mockLoadFromFile(suarestexture, "app/players/suares.jpg");
    mockLoadFromFile(zlatantexture, "app/players/zlatan.jpg");

    oblaksprite.setTexture(oblaktexture);
    suaressprite.setTexture(suarestexture);
    zlatansprite.setTexture(zlatantexture);
    messiprite.setTexture(messitexture);
    mbappesprite.setTexture(mbappetexture);
    noyersprite.setTexture(noyertexture);
    neymarsprite.setTexture(neymartexture);
    akinfeevsprite.setTexture(akinfeevtexture);
    bufonsprite.setTexture(bufontexture);
    modricsprite.setTexture(modrictexture);
}

void mockClubLib()
{
    mockLoadFromFile(BarcelonaT, "app/club/barcelona.png");
    mockLoadFromFile(BavariaT, "app/club/bayern-munich.png");
    mockLoadFromFile(BorissaT, "app/club/borussia.png");
    mockLoadFromFile(KobenhavnT, "app/club/kobenhavn.png");
    mockLoadFromFile(qingbaoT, "app/club/qingdao.png");
    mockLoadFromFile(RealT, "app/club/real-madrid.png");
    mockLoadFromFile(salzburgT, "app/club/salzburg.png");
    mockLoadFromFile(santosT, "app/club/santos.png");
    mockLoadFromFile(spartakT, "app/club/spartak.png");
    mockLoadFromFile(torontoT, "app/club/toronto.png");

    BarcelonaS.setTexture(BarcelonaT);
    BavariaS.setTexture(BavariaT);
    BorissaS.setTexture(BorissaT);
    KobenhavnS.setTexture(KobenhavnT);
    qingbaoS.setTexture(qingbaoT);
    RealS.setTexture(RealT);
    salzburgS.setTexture(salzburgT);
    santosS.setTexture(santosT);
    spartakS.setTexture(spartakT);
    torontoS.setTexture(torontoT);
}

TEST(WinFunctionTest, WinCondition)
{
    poswin = 2;
    checkwin = 2;
    ret = 0;
    kol_vo = 0;

    int result = win();
    ASSERT_EQ(result, 2);
    ASSERT_EQ(kol_vo, 1);
}

TEST(WinFunctionTest, IncorrectAnswer)
{
    poswin = 1;
    checkwin = 3;
    ret = 0;
    kol_vo = 0;

    int result = win();
    ASSERT_EQ(result, 1);
    ASSERT_EQ(kol_vo, 0);
}

TEST(WinFunctionTest, InvalidInput)
{
    poswin = 100;
    checkwin = 2;
    ret = 0;
    kol_vo = 0;

    int result = win();
    ASSERT_EQ(result, 0);
    ASSERT_EQ(kol_vo, 0);
}

TEST(PlayerLibTest, TexturesLoadCorrectly)
{
    playerlib();
    ASSERT_TRUE(messitexture.getSize().x > 0);
    ASSERT_TRUE(mbappetexture.getSize().x > 0);
    ASSERT_TRUE(noyertexture.getSize().x > 0);
    ASSERT_TRUE(neymartexture.getSize().x > 0);
    ASSERT_TRUE(akinfeevtexture.getSize().x > 0);
    ASSERT_TRUE(bufontexture.getSize().x > 0);
    ASSERT_TRUE(modrictexture.getSize().x > 0);
    ASSERT_TRUE(oblaktexture.getSize().x > 0);
    ASSERT_TRUE(suarestexture.getSize().x > 0);
    ASSERT_TRUE(zlatantexture.getSize().x > 0);
}

TEST(ClubLibTest, TexturesLoadCorrectly)
```

```
{
    clublib();
    ASSERT_TRUE(BarcelonaT.getSize().x > 0);
    ASSERT_TRUE(BavariaT.getSize().x > 0);
    ASSERT_TRUE(BorissaT.getSize().x > 0);
    ASSERT_TRUE(KobenhavnT.getSize().x > 0);
    ASSERT_TRUE(qingbaoT.getSize().x > 0);
    ASSERT_TRUE(RealT.getSize().x > 0);
    ASSERT_TRUE(salzburgT.getSize().x > 0);
    ASSERT_TRUE(santosT.getSize().x > 0);
    ASSERT_TRUE(spartakT.getSize().x > 0);
    ASSERT_TRUE(torontoT.getSize().x > 0);
}

TEST(QuizplayerTest, ValidTruePlayer)
{
    srand(static_cast<unsigned int>(time(nullptr)));
    Quizplayer();
    ASSERT_TRUE(checkwin >= 0 && checkwin < 4);
    ASSERT_TRUE(positions[checkwin] == 2);
    ASSERT_TRUE(posnames[checkwin] == names[TruePlayer]);
}

TEST(QuizplayerTest, ValidFalsePlayers)
{
    srand(time(nullptr));
    Quizplayer();
    for (int i = 0; i < 4; ++i) {
        if (i != checkwin) {
            ASSERT_NE(posnames[i], names[TruePlayer]);
        }
    }
}

TEST(LibraryTest, PlayerLibTexturesLoad)
{
    mockPlayerLib();

    // Проверка установки текстур
    EXPECT_EQ(messiprite.getTexture(), &messitexture);
    EXPECT_EQ(mbappesprite.getTexture(), &mbappetexture);
    EXPECT_EQ(noyersprite.getTexture(), &noyertexture);
    EXPECT_EQ(neymarsprite.getTexture(), &neymartexture);
    EXPECT_EQ(akinfeevsprite.getTexture(), &akinfeevtexture);
    EXPECT_EQ(bufonsprite.getTexture(), &bufontexture);
    EXPECT_EQ(modricsprite.getTexture(), &modrictexture);
    EXPECT_EQ(oblaksprite.getTexture(), &oblaktexture);
    EXPECT_EQ(suaressprite.getTexture(), &suarestexture);
    EXPECT_EQ(zlatansprite.getTexture(), &zlatantexture);
}

TEST(LibraryTest, ClubLibTexturesLoad)
{
    mockClubLib();

    // Проверка установки текстур
    EXPECT_EQ(BarcelonaS.getTexture(), &BarcelonaT);
    EXPECT_EQ(BavariaS.getTexture(), &BavariaT);
    EXPECT_EQ(BorissaS.getTexture(), &BorissaT);
    EXPECT_EQ(KobenhavnS.getTexture(), &KobenhavnT);
    EXPECT_EQ(qingbaoS.getTexture(), &qingbaoT);
    EXPECT_EQ(RealS.getTexture(), &RealT);
    EXPECT_EQ(salzburgS.getTexture(), &salzburgT);
    EXPECT_EQ(santosS.getTexture(), &santosT);
    EXPECT_EQ(spartakS.getTexture(), &spartakT);
    EXPECT_EQ(torontoS.getTexture(), &torontoT);
}
```